UR5 Collision Avoidance
An Annotated Bibliography

Saym Imtiaz (simtiaz@wisc.edu)
University of Wisconsin-Madison

August 25, 2022

References

[1] "Clustering," https://scikit-learn.org/stable/modules/clustering.html, accessed: 2022-7-26.

> This resource summarizes the different clustering algorithms implemented in scikit. It also gave performance comparisons and use cases for each method, which was very helpful in selecting k-means, minibatch k-means, and DBSCAN for this project.

[2] "Compare BIRCH and MiniBatchKMeans," https://scikit-learn.org/stable/auto_examples/cluster/plot_birch_vs_minibatchkmeans.html#sphx-glr-auto-examples-cluster-plot-birch-vs-minibatchkmeans-py, accessed: 2022-7-26.

> This article demonstrated that Mini-Batch K-Means runs considerably faster than Birch with similar clustering results. This was useful in deciding to use Mini-Batch K-Means over Birch for the clustering method.

[3] "Comparison of the K-Means and MiniBatchKMeans clustering algorithms," https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html#sphx-glr-auto-examples-cluster-plot-mini-batch-kmeans-py, accessed: 2022-7-26.

This article demonstrates that Mini-Batch K-Means is both faster than K-Means and incurs negligible clustering penalties. It therefore was determined to be a better choice for clustering objects for this project.

[4] "Downsampling a PointCloud using a VoxelGrid filter," https://pcl.readthedocs.io/projects/tutorials/en/latest/voxel_grid.html, accessed: 2022-7-26.

I used this source to downsample the raw point cloud input. It uses a voxelized grid approach by approximating each voxel by the centroid of the enclosed points. This reduces the amount of points often by an order of magnitude and thus considerably speeds up the collision object extraction.

[5] "Euclidean Cluster Extraction," https://pcl.readthedocs.io/projects/tutorials/en/master/cluster_extraction.html#cluster-extraction, accessed: 2022-7-26.

This tutorial demonstrates how to perform DBSCAN using Point Cloud Library on a point cloud. However, I used scikit's implementation of DBSCAN as this implementation removed planar models from the clustering. This is undesirable behavior for this project as it is important to model all surfaces and objects in CollisionIK for collision avoidance.

[6] "Fitting a plane to noisy points in 3D," https://www.ilikebigbits.com/2017_09_25_plane_from_points_2.html, accessed: 2022-7-26.

A source I explored in trying learn plane-fitting to create oriented bounding boxes.

[7] "How to incrementally register pairs of clouds," https://pcl.readthedocs.io/projects/tutorials/en/latest/pairwise_incremental_registration.html#pairwise-incremental-registration, accessed: 2022-7-26.

This document helped show how the Iterative Closest Point algorithm could be used with the Point Cloud Library to incrementally register point clouds. It is a helpful source for creating a point cloud scan. It would be especially useful for trying to create a scan of an environment from a fixed tripod after calibrating the camera position to the robot position.

[8] "Introductory Guide to AABB Tree Collision Detection," https://www.azurefromthetrenches.com/introductory-guide-to-aabb-tree-collision-detection/, accessed: 2022-7-26.

This resource helped to understand how AABB trees work for collision detection. It also explained the algorithm for creating, updating, and searching the tree. However, I did not use this resource directly as it would require modifying the CollisionIK code since it currently handles each AABB seperately. Implementing AABB tree support, or even OBB tree support could speed up CollisionIK as it currently queries every object within a certain radius of each linkage.

[9] "K-Means++ Algorithm for High-Dimensional Data Clustering," https://towardsdatascience.com/k-means-algorithm-for-high-dimensional-data-clustering-714c6980daa9, accessed: 2022-7-26.

This resource summarizes both traditional K-Means with the Lloyd-Forgy algorithm as well as the K-Means++ algorithm. It also shows how the K-Means++ algorithm both has reduced runtime and early convergence, and thus led me to select K-Means++ when using the scikit implementation.

[10] "K-Means Clustering in Python: A Practical Guide," https://realpython.com/k-means-clustering-python/, accessed: 2022-7-26.

This resource was useful in understanding the different classes of clustering algorithms and their respective benefits. It also talks about how to select the k coefficient for K-means such as the elbow method and silhouette coefficient. However, there does not seem to be a good k coefficient selector based on my use case of trying to break down objects into a collection of axis-aligned bounding boxes.

[11] "Online learning of a dictionary of parts of faces," https://scikit-learn.org/stable/auto_examples/cluster/plot_dict_face_patches.html#sphx-glr-auto-examples-cluster-plot-dict-face-patches-py, accessed: 2022-7-26.

This tutorial shows how to use Mini-Batch K-Means online to continuously update clusters. It is a potential avenue to using clustering algorithms in real-time as the tutorial demon-

strates considerably fast speeds, achieving a 0.9 second run time on 3200 image patches.

[12] "Removing outliers using a Conditional or RadiusOutlier removal," https://pcl.readthedocs.io/projects/tutorials/en/master/remove_outliers.html#remove-outliers, accessed: 2022-7-26.

While I did not directly use the software present in this tutorial, it did inspire filtering out point cloud data that was captured a certain radius from the UR5 base. This intuitively makes sense as the UR5 cannot collide with obstacles further than its operating volume.

[13] "sklearn.cluster.MiniBatchKMeans," https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans, accessed: 2022-7-26.

This is the documentation for scikit's Mini-Batch K-Means which is utilized extensively in this project.

[14] "Using a matrix to transform a point cloud," https://pcl.readthedocs.io/projects/tutorials/en/master/matrix_transform.html#matrix-transform, accessed: 2022-7-26.

This source helped me understand how to apply a transform to a point cloud. It was helpful in debugging issues I had in calibration and helped me realize the real issue is that CollisionIK's ROS transform tree does not match the Universal Robots' transform tree.

[15] "The URScript Programming Language," https://s3-eu-west-1.amazonaws.com/ur-support-site/32554/scriptManual-3.5.4.pdf, April 2018, accessed: 2022-7-26.

This source helped me figure out the UR commands to move the UR5 with CollisionIK.

[16] P. Chanapol, "Rospointcloudtransformer," https://github.com/petpetpeter/ROS-point-cloud-Transformer, November 2021.

This software package was used to transform the point cloud data from the camera frame to the robot base frame.

[17] S. Dinas and J. Bañón, "A literature review of bounding volumes hierarchy focused on collision detection," vol. 17, pp. 49–62, 06 2015.

This source summarizes different bounding box techniques for collision detection. It helped me look at different technqiues that may be easily compatible with CollisionIK such as axis-aligned bounding boxes, oriented-bounding boxes, spheres, meshes, and potentially ellipsoids. Given my use case of static obstacles but maintaining fast collision checking, I decided to use axis-aligned bounding boxes. I also chose boxes over spheres because while spheres are invariant to rotation and translations, these advantages are not very useful given their trade-off of having low volume tightness.

[18] M. P. Eitan Marder-Eppstein, Tomas Petricek, "Robot body filter." [Online]. Available: http://wiki.ros.org/robot_body_filter

This is useful source implemented in ROS for how to filter out the UR5 from point cloud scans when the camera scan is not done on the end-effector. While I did not end up using this as I opted for a camera mount, it could be quite helpful in the future for tripod scans. Furthermore, while the intended use is to filter out points that correspond to the robot, it is most likely possible to extend this to already scanned static obstacles. This could then entail filtering out all points that correspond to static obstacles which could speed up update times when handling dynamic obstacles.

[19] F. Exner, "Universal_robots_ros_driver," https://github.com/UniversalRobots/Universal_Robots_ROS_Driver, 2022.

This is the software package that the calibration package uses to move the UR5 and keep track of the reference frames at each measurement in order to calculate the transformation from "eye" to robot.

[20] M. H. Fjeld, "3d spatial navigation in octrees with reinforcement learning," Master's thesis, University of Oslo, 2018.

This master's thesis was primarily useful for giving me direction on how to write my project report and how to structure the paper itself. It also was helpful in understanding voxel

and OctoMaps in a more digestible manner. It also supports the claim that octrees can be used for 3D navigation and collision avoidance. The author of this paper also does note that raycasting occupancy mapping is a promising direction of research similar to what RCIK uses.

[21] D. Fridovich-Keil, E. Nelson, and A. Zakhor, "Atommap: A probabilistic amorphous 3d map representation for robotics and surface reconstruction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3110–3117.

This is an occupancy grid 3D mapping technique similar to OctoMap. It uses spheres instead of voxels in order to reduce quantization effects and more accurately preserve the continuous nature of surfaces. It's a useful source as it could potentially be used with CollisionIK to more accurately model the environment compared to bounding volume hierarchies. It also is faster to update compared to OctoMap, but still at only about 10-20Hz.

[22] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, 04 2013.

This is a popular voxel-based 3D mapping technique that includes both free and unknown areas. It also use octrees which enable it to be incredibly memory efficient in storing map information, and has support to be generated directly from point cloud data. However, it is not quite real-time in that it runs at about 10-20Hz.

[23] M. Kang, Y. Cho, and S.-E. Yoon, "RCIK: Real-Time Collision-Free Inverse Kinematics Using a Collision-Cost Prediction Network," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 610–617, 2022.

The authors of this paper present real-time collision-free inverse kinematics on a 6DOF robot for both static and dynamic obstacles. It differs from CollsionIK most notably for their use of a collision-cost prediction network and using an occupancy grid instead of mesh objects. However, while it lends itself well to working with real-time sensor data, it does struggle to scale for higher DOF robots.

[24] Y. Kwon, D. Kim, I. An, and S.-e. Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 482–497, 2019.

This is the method used by RCIK to solve the issue of occupancy grids typically taking too long to update in real-time. This could be a great method to try with CollisionIK in the future as it able to achieve real time speeds in indoor environments. For a resolution of of 0.2m, it achieves speeds of 18-23 FPS whereas for 0.4m this jumps up to 54.5-69 fps.

[25] T. L. Lam, H. W. Yip, H. Qian, and Y. Xu, "Collision avoidance of industrial robot arms using an invisible sensitive skin," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4542–4543.

Related work different from CollsionIK in that it uses a sensor skin in order to avoid obstacles.

[26] G. C. Lei Zhang, "ur5_realsense_calibration," https://github.com/portgasray/ur5_realsense_calibration, 2020.

This is the calibration package I used.

[27] F. Poux, "3D Point Cloud Clustering Tutorial with K-means and Python," https://towardsdatascience.com/3d-point-cloud-clustering-tutorial-with-k-means-and-python-c870089f3af8, accessed: 2022-7-26.

This tutorial walks through the process of taking a point cloud scan of an airport and extracting objects using DBSCAN and then K-Means clustering. It was both useful in seeing how to code the scikit implementation of these clustering algorithms but also clearly showed the advantage of combining DBSCAN with another clustering algorithm. This is where I got the idea to use DBSCAN with Mini-Batch K-means for the project.

[28] D. Rakita, H. Shi, B. Mutlu, and M. Gleicher, "Collisionik: A per-instant pose optimization method for generating robot motions with environment collision avoidance," *CoRR*, vol. abs/2102.13187, 2021. [Online]. Available: https://arxiv.org/abs/2102.13187

This is the basis for generating robot motion when given environmental data and real-time motion objectives. As such, it was critical to understand this paper to understand what kind of environmental data to pass into so that it would be able to generate real-time motion. CIK thus led to my decision to opt for axis-aligned bounding boxes. It also led to my decision to try and minimize the number of boxes total and to not pursue OctoMaps or AtomMap.

[29] H. Schmidl, N. Walker, and M. Lin, "Cab: Fast update of obb trees for coillision detection between articulated bodies," *Journal of Graphics Tools*, vol. 9, 01 2004.

This paper discusses an approach of using a oriented bounding box hierarchy for articulated humanoid models. The authors are able to achieve considerable speedup in update time and thus total collision checking time by implementing their method for scenarios such as a hand interacting with a ball. This source is useful as it could potentially be a great way on handling other robots or human obstacles in a dynamic environment and would be compatible with CollisionIK.

[30] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

This is the base technique utilized for calibration.

[31] W. M. Tully Foote, Eitan Marder-Eppstein, "tf2." [Online]. Available: http://wiki.ros.org/tf2

This is how ROS is able to keep track of different reference frames and how they are related to each other. This source was very useful in debugging my calibration issues and creating the linkage between the camera mount and the robot base for point cloud transformations.

[32] S. Vanneste, B. Bellekens, and M. Weyn, "3dvfh+: Real-time three-dimensional obstacle avoidance using an octomap," vol. 1319, 07 2014.

This paper utilizes OctoMap for 3D obstacle avoidance. It was an example that demonstrates occupancy grid mapping can be used for environment collision avoidance and led me

to explore OctoMaps for a little bit. However, their method is only applicable to already generated environment mapping and cannot handle dynamic obstacles.

[33] L. Zhao, J. Zhao, H. Liu, and D. Manocha, "Collision-free kinematics for redundant manipulators in dynamic scenes using optimal reciprocal velocity obstacles," *CoRR*, vol. abs/1811.00600, 2018. [Online]. Available: http://arxiv.org/abs/1811.00600

Related work in collision avoidance where the robot can avoid self-collision and collisions with other robots by modeling linkages as "velocity objects". It does not seek to address the problem of environmental collision avoidance however.